April 2011

A Particle-based Method for the Simulation of Complex Fluids and Polymer Solutions

James Leonard Kingsley
Worcester Polytechnic Institute

Project Number: MQP _ ERT _ JK10

A PARTICLE-BASED METHOD FOR THE SIMULATION OF COMPLEX FLUIDS AND POLYMER SOLUTIONS

Major Qualifying Project Report completed in partial fulfillment
of the Bachelor of Science degree at
Worcester Polytechnic Institute

James Kingsley

April 28, 2011

Professor Erkan Tüzel, Advisor
Department of Physics

Abstract

Stochastic Rotation Dynamics (SRD) is a particle-based simulation method that can be used to model fluids. It has many of the benefits of direct simulation methods, while still being computationally inexpensive. With hydrodynamic interactions and thermal fluctuations built-in, it can be used in mesoscale simulations of complex fluids. Starting with a simple, ideal fluid, the method was verified for theoretical consistency and extended to support non-ideal equations of state. As expected, the non-ideal fluid demonstrated freezing behavior at low temperatures. It has been shown that by employing these repulsive interactions, the model can be applied to binary mixtures. Here, we demonstrate this technique by studying the phase separation of two fluids and the formation of droplets.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computational fluid simulations are useful for many applications, and there are a number of different techniques for it. However, for many of them, it is difficult to include arbitrary structures in the fluid, such as polymers, or model multiple types of fluids with thermal fluctuations.

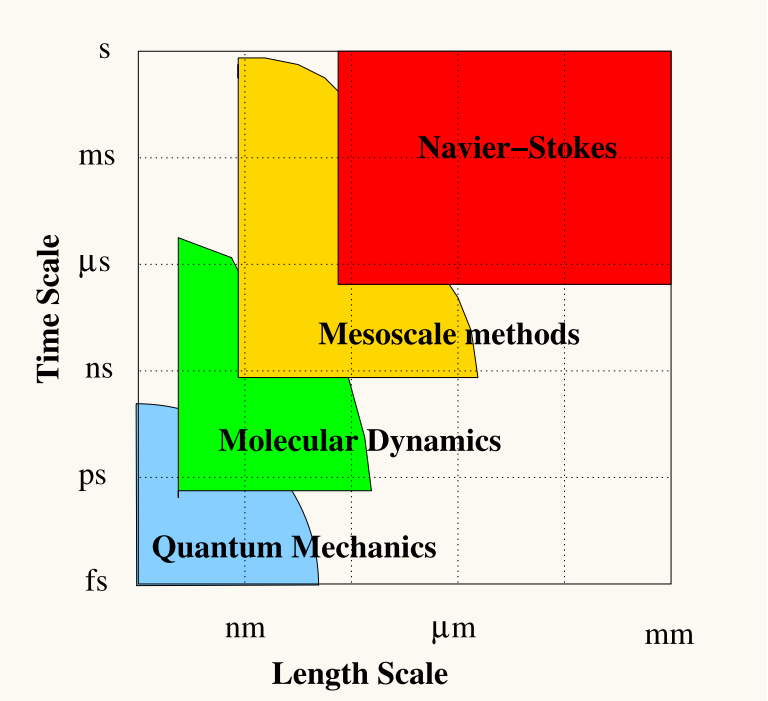


Figure 1.1: Different length and time scales that span various simulation methods.

In general, fluid simulations can be created from the top down or from the bottom up. The method chosen depends on the requirements of the specific situation. The length and time scales of applicability of various simulation methods are shown in Figure 1.1.

1.1 Top Down Approach

Top down simulations typically start with differential equations for the behavior of fluids, and simulate the fluid as a continuous medium. If the domain under consideration is small enough that the fluid is no longer entirely continuous, it does not correctly model it. This type of method is appropriate for many types of fluid but it usually works best at a scale large enough for the fluid to behave deterministically. There are two popular approaches: the Navier-Stokes solvers and lattice methods. In what follows, we will briefly review these methods.

1.1.1 Navier-Stokes Methods

The Navier-Stokes equations can be solved numerically in various ways. The finite volume method breaks the experimental domain into a number of small, usually triangular subdomains. Each subdomain is a controlled volume element, and is treated as a homogeneous entity. The starting point for the simulation is a set of equations describing the behavior of the fluid. Usually, continuum mass and momentum equations are used. Using the divergence theorem, any divergence terms can be converted into surface integrals, which are just transport across boundaries to neighboring subdomains. From there, as expected, a system of ordinary differential equations representing the behavior of the system can be generated. This system of differential equations can then be numerically solved using methods such as Euler's method or Runge-Kutta.[8]

1.1.2 Lattice Methods

The first of the lattice methods was the lattice gas automata [9]. In lattice gas automata, there are a series of particles which move along a discrete lattice. When they colocate on a single point in the lattice, collision rules apply. With appropriate choice of collision rules that conserve momentum and energy, accurate hydrodynamic behavior emerges at the macroscopic scale. The lattice requires that both position and velocity are discretized, which means that the simulation is inherently noisy.

An improvement on this is the lattice Boltzmann model [12, 18, 20]. This method solves the discrete Boltzmann equation over a discrete lattice mesh. The improvement over lattice gas automata is that lattice Boltzmann uses particle probability densities instead of entire particles. This means that position and velocity can be approximately continuous. Even so, both methods suffer from Galilean invariance problems, since they both are on a discrete grid.

1.2 Bottom Up Approach

Alternatively, bottom-up methods can be used to model the behavior of fluids. In this case, the lowest-level physical rules are combined and applied, giving rise to the macroscopic behaviors of fluids. Below, we briefly describe some examples of bottom up techniques.

1.2.1 Quantum Simulation

Starting at the very bottom, there is quantum-mechanical simulations[3, 17]. The most powerful and expensive simulation type, it involves calculation and direct manipulation of wavefunctions. This is often done by solving Schrödinger's equation with some amount of approximation[4]. This provides the most complete and accurate description of a fluid, but in most cases is prohibitively expensive. Additionally, there are multiple methods of taking the approximations, and choosing the appropriate one for a given situation is difficult. As such, a full quantum mechanical model is often only used in cases where quantum effects are known to be important to the behavior of a system.

1.2.2 Molecular Dynamics

To avoid having to solve the Schrödinger equation, molecular dynamics simulations treat the situation classically. Each individual atom in the system is modeled using Newton's laws under the influence of given potentials, without including quantum mechanical effects. Instead of wavefunction interactions, interactions are all done as inter-particle potentials. These potentials usually are electrostatic, though they can in theory be anything, such as Lennard-Jones, etc. This is significantly less expensive than a quantum-mechanical simulation, but still is untenable for very large systems. Current supercomputing systems can handle molecular dynamics simulations for hundreds of thousands of atoms, but that is still too little for many applications. Molecular dynamics is best for modeling complex atomic things like proteins, in which case a more coarse-grained model cannot simulate the system properly. However, for cases such as large volumes of a simple solvent, this is unnecessarily complex and time consuming.

1.3 Mesoscale Models

Mesoscale models have some similar properties to molecular dynamics models, but they use particles which represent a region of the fluid. They do not resolve the full microscopic details, as in molecular dynamics, but they still have fluctuations and produce correct hydrodynamics hydrodynamic behavior. In what follows, we will give a few examples of such models.

1.3.1 Dissipative Particle Dynamics

In order to avoid some of the artifacts inducted by lattice models, dissipative particle dynamics was developed[6, 7, 13]. It is a particle-based model, with each particle representing a fluid region. The interactions between individual particle pairs are a sum of three forces. The first interaction force is a conservative one. This force, as per its name, conserves energy as the particles interact, and is responsible for most of the behavior of the fluid. The dissipative force normalizes energy in the system. This allows the entire system to be kept close to a constant temperature. The last force is the random force. This is what gives rise to random behavior, and is what makes this method not deterministic. All of the forces are all antisymmetric as per Newton's third law, which means that momentum is conserved in

interactions. While the conservative force does conserve energy, the dissipative and random forces, do not. On average, however, the system stays at a constant (set) temperature.

1.3.2 Direct Simulation Monte Carlo

Direct Simulation Monte Carlo (DSMC) is the oldest of the mesocale models[1, 2, 10]. It models fluids using aggregate representative particles. These particles, instead of modeling a single atom or simple molecule, are representative of an entire region of the fluid. The model runs in discrete timesteps with continuous position and velocity vectors. During each timestep, the particles undergo a streaming and a collision step. In the streaming step, the particles advance their positions through time ballistically. In the collision step, the particles collide with each other via a set of collision rules. The system is divided into a set of boxes and only particles within a given box have a chance to collide. These collisions are calculated using probabilistic models, with collisions happening between individual particle pairs. Since collisions are elastic (unless intentionally set otherwise), DSMC conserves energy and momentum.

1.3.3 Stochastic Rotation Dynamics

Stochastic Rotation Dynamics (SRD) is very similar to DSMC, but it uses a different collision scheme [11, 14, 16]. It still has the same type of particles, and the same streaming and collision steps. The difference lies in the method by which the collision step is implemented. Instead of having randomly chosen particle pairs collide with each other, it collides groups of particles with each other simultaneously. These groups are picked by dividing the simulation domain into a set of boxes. All the particles within a box collide with each other at once, using the simplest available collision rule that conserves momentum and energy. Since particle pairs are not considered, the simulation scales much better than DSMC. This means that SRD can simulate larger systems more efficiently.

1.3.4 Dimensionless Parameters

Due to the mesoscale nature of these simulations, directly setting parameters based on real systems does not adequately set the behavior of a system. This is due to the unphysical nature of the units used in the simulation. While quantum mechanical and molecular dynamics simulations can have parameters with physical meaning, the aggregate pseudo-particles used in these mesoscale methods have no direct physical significance. It is necessary to represent the system with dimensionless parameters. As dimensionless parameters, their value is independent of the dimensions of the fluid they are in, and as such can be used to compare the behaviors of different fluids. Examples of commonly used dimensionless numbers are the Reynolds, Schmitt, Peclet, and Prandtl numbers.

The Reynolds number is a measure of the ratio of inertial forces to viscous forces in a fluid [5]. In low Reynolds number fluids, viscous forces dominate, and turbulence is minimal as a result. In high Reynolds number fluids, however, inertial forces are more significant.

The Schmidt number measures the ratio of momentum transport to mass transport [19]. For fluids like water, it is on the order of $10^2 - 10^3$, while for gases it is close to 1.

The Peclet number is the ratio of transport of a quantity to the diffusion of the same quantity [15]. The Peclet number is the product of the Reynolds and Prandtl numbers.

The Prandtl number is the ratio of kinematic viscosity to thermal diffusivity[22].

1.4 Summary

There are a number of methods available for simulating fluids. At one end of the spectrum there are continuum methods such as Navier-Stokes solvers. These are very good for large-scale simulation of macroscopic situations. This simulation type is commonly used in industry to model fluid flows in and around systems ranging from airplanes to windmills. However, they are difficult to implement for small scales. Lattice methods are better at handling fluctuations, but also can suffer from some instabilities due to discrete positions and velocities. This is particularly an issue in the case of multi-component mixtures. At the other end of the granularity spectrum, there are high-accuracy atomic simulations using quantum mechanics or molecular dynamics. These are very expensive and used in research applications to get results that no other method is capable of providing. In the following chapters, we will focus on the coarse grained method SRD and its extensions.

Chapter 2

Stochastic Rotation Dynamics

The SRD simulation method was originally proposed by Malevanets and Kapral in 1999[16]. As briefly described earlier, it uses a set of particles, which move around and collide with each other in a coarse-grained fashion. This is easy and fast to simulate, and has the advantage of making it easy to include other objects into the simulation, since the interaction between a particle and an arbitrary object is comparatively simple to simulate. The mechanics of the SRD method are as follows.

2.1 Description of Model

In a typical SRD simulation, there are N particles on a field of side length L. The field is divided into boxes of side length a, as shown in Figure 2.1. This gives an average of

$$M = \frac{Na^2}{L^2} \tag{2.1}$$

particles per cell. Each particle has a mass m, a position $\vec{r}$, and a velocity $\vec{v}$.

The SRD method has two main parts that it performs in each timestep. The first step is the streaming step, where particles increment forward in time. At this point, boundary conditions are applied, and any additional forces are added. This is followed by the collision step, where the particles collide with each other, as described in Section 2.2.2.

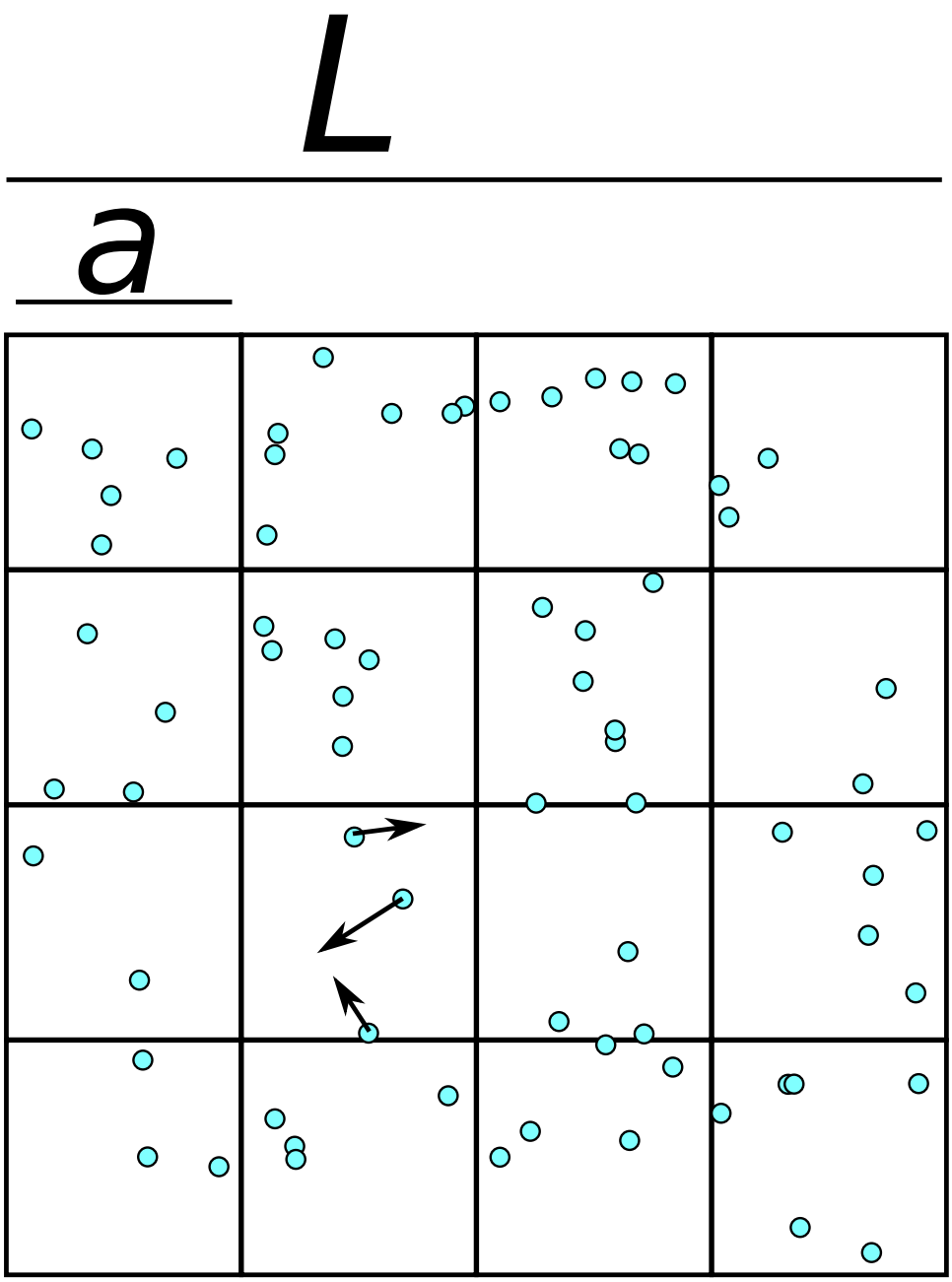


Figure 2.1: Example SRD grid with particles.

2.2 Dynamics

2.2.1 Streaming

The first part of the core simulation is the simple streaming step. The particles take a step through time, advancing their positions. Note that for notational convenience, when referring to the properties of a single particle, the subscript will be dropped, since the equation applies to all particles individually . In its most basic form, streaming is done with a simple Euler integration,

$$\vec{x}(t+\tau) = \vec{x}(t) + \vec{v}(t+\tau)\tau \quad , \tag{2.2}$$

where t is the time, and τ is the timestep duration.

During the streaming step (depending on the effect, applied before, instead of, or after, as appropriate), additional forces can be applied to the particles. For example, a gravitational acceleration term could be applied to all particles, to create a gravity-driven flow.

2.2.2 Collision

The streaming step is followed by the collision step. In order to collide the particles, a grid of boxes is placed across the field. All of the particles are assigned into whichever box they are in, and total momentum is summed for each box. The collision is a rotation of the velocities of all the particles in each box, relative to the overall momentum of the box. This is done as a matrix multiplication of a rotation matrix $\boldsymbol{R}$ for an angle α on the relative velocity vector $\vec{v} - \vec{u}$, and given by

$$\vec{v}(t+\tau) = \mathbf{R}\left(\vec{v}(t) - \vec{u}(t)\right) + \vec{u}(t) \quad , \tag{2.3}$$

where $\vec{u}$ is given by

$$\vec{u} = \frac{\sum_{i=1}^{M'} m_i \vec{v}_i}{m_t} \quad . \tag{2.4}$$

M' is the number of particles in a box at any given time. On average, $\langle M' \rangle = M$. Equation (2.4) also uses the total mass in the box, i.e.,

$$m_t = \sum_{i=1}^{M'} m_i \quad . \tag{2.5}$$

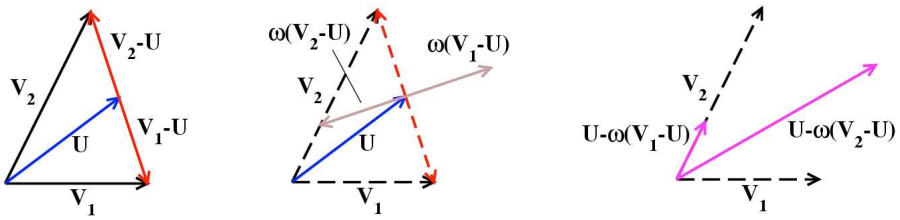
$\mathbf{R}$ is given by

$$\boldsymbol{R} = \begin{pmatrix} \cos(\alpha) & \pm\sin(\alpha) \\ \mp\sin(\alpha) & \cos(\alpha) \end{pmatrix} \quad . \tag{2.6}$$

Since the rotation matrix $\mathbf{R}$ is an orthogonal matrix, it satisfies

$$\boldsymbol{R}\boldsymbol{R}^{-1} = \boldsymbol{R}\boldsymbol{R}^{\mathrm{T}} = \boldsymbol{I} \quad . \tag{2.7}$$

In order to ensure detailed balance, the direction of rotation is randomly picked for each box. This is implemented by randomly picking a plus or minus sign in Equation (2.6). The collision process is demonstrated for two particles in Figure 2.2.2.



(a) Particles come in with velocities $\vec{V}_1$ and $\vec{V}_2$

(b) Relative velocities are rotated by α

(c) Initial and final velocities

Figure 2.2: Example of an SRD collision with two equal mass particles. The collision angle $\alpha = 90°$, and U is the mean velocity of the two particles.

2.2.3 Energy and Momentum Conservation

Since the only operation being applied to the particles is a rotation in a static frame, kinetic energy and momentum are conserved within each box. This can be shown by explicitly calculating the momentum before and after the collision

$$\sum_{i=1}^{M} m_i \vec{v}_i(t+\tau) \;=\; \sum_{i=1}^{M} m_i \mathbf{R}\left(\vec{v}_i(t) - \vec{u}(t)\right) + m_t \vec{u}(t) \quad .$$

Using the definition of $\vec{u}$ from Equation (2.4) gives

$$\begin{aligned}
\sum_{i=1}^{M'} m_i \vec{v}_i(t+\tau) &= m_t \vec{u}(t) + \sum_{i=1}^{M'} \mathbf{R} m_i \left(\vec{v}_i(t) - \vec{u}(t)\right) \\
&= m_t \vec{u}(t) + \sum_{i=1}^{M'} \mathbf{R} m_i \left(\vec{v}_i(t)\right) - m_t \mathbf{R} \vec{u}(t) \\
&= m_t \vec{u}(t) + \sum_{i=1}^{M'} \mathbf{R} m_i \left(\vec{v}_i(t)\right) - \sum_{i=1}^{M'} m_i \mathbf{R} \vec{v}_i(t) \\
&= \sum_{i=1}^{M'} m_i \vec{v}_i(t) + \sum_{i=1}^{M'} m_i (\mathbf{R}\left(\vec{v}_i(t) - \vec{v}_i(t)\right) \\
&= \sum_{i=1}^{M'} m_i \vec{v}_i(t) \quad .
\end{aligned}$$

Energy conservation can similarly be calculated by

$$\begin{aligned}\sum_{i=1}^{M}\frac{1}{2}m_i\left|\vec{v}_i(t+\tau)\right|^2 &= \sum_{i=1}^{M'}\frac{1}{2}m_i\left(\mathbf{R}\left(\vec{v}_i(t)-\vec{u}(t)\right)+\vec{u}(t)\right)^2 \\ &= \sum_{i=1}^{M'}\frac{1}{2}m_i\left(\vec{u}(t)^2+2\mathbf{R}\left(\vec{v}_i(t)-\vec{u}(t)\right)\vec{u}(t)+\mathbf{R}\mathbf{R}^{\mathrm{T}}\left(\vec{v}_i(t)-\vec{u}(t)\right)^2\right)\end{aligned}$$

which by Equation (2.7) becomes

$$\sum_{i=1}^{M'}\frac{1}{2}m_i\left|\vec{v}_i(t+\tau)\right|^2 = \sum_{i=1}^{M'}\frac{1}{2}m_i\left(\vec{u}(t)^2+2\mathbf{R}\left(\vec{v}_i(t)-\vec{u}(t)\right)\vec{u}(t)+\left(\vec{v}_i(t)-\vec{u}(t)\right)^2\right) \quad ,$$

and the expansion of the last term, by Equation (2.4) becomes

$$\begin{aligned}\sum_{i=1}^{M'}\frac{1}{2}m_i\left|\vec{v}_i(t+\tau)\right|^2 &= \frac{1}{2}m_t\left(2\vec{u}(t)^2-2\vec{u}(t)^2+2\mathbf{R}\left(\vec{u}(t)-\vec{u}(t)\right)\vec{u}(t)\right)+\sum_{i=1}^{M'}\frac{1}{2}m_i\vec{v}_i(t)^2 \\ &= \sum_{i=1}^{M'}\frac{1}{2}m_i\vec{v}_i(t)^2 \quad .\end{aligned}$$

Since momentum and energy are conserved within each individual box, they are also conserved in the sum of all boxes.

2.2.4 Shifting

In cases where a simulation includes an arbitrarily imposed structure, there is a potential for Galilean invariance problems. In SRD, diffusive transport is affected (see Section 2.5.4)[14]. When there is a flow in one direction (or equivalently, the reference frame is moving), diffusion in the direction of the flow is faster than it would otherwise be (Figure 2.15). In order to resolve this, the location of the whole grid is randomly shifted at each timestep. Instead of every collision happening with the boxes in the same place, each timestep places them in a different location, offset randomly in the range $[-a/2, a/2]$. Since the range is equal to the size of the boxes, this shifting means that the boxes are "blurred" across their entire size. An example shift by a vector $\vec{b}$ is given in Figure 2.3.

Even with shifting, the simulation still has a square structure to it. This shows up as a higher order effect such as in the freezing transition demonstrated in Section 4.2.2.

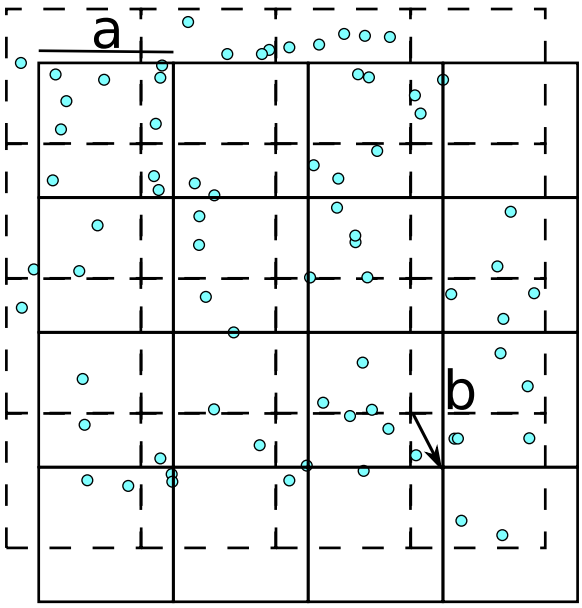


Figure 2.3: A collision grid of side length a that is shifted by $\vec{b}$, overlaid on its unshifted position.

2.3 Initialization

In a typical simulation, the particles are initialized with positions drawn from a uniform distribution across the entire field. Their velocities are also drawn from a uniform distribution, but a uniform distribution does not guarantee zero overall momentum and a specific energy or temperature. As such, the starting distribution is arbitrary and irrelevant as long as it is uniform, because the velocities of all particles are then scaled to match the specified momentum and temperature. First, the momentum is normalized by taking the average velocity of the system

$$\vec{v}_{total} = \frac{\sum_{i=1}^{N} m_i \vec{v}_i}{\sum_{i=1}^{N} m_i} \tag{2.8}$$

and subtracting it from the velocity of each individual particle. This velocity is similar to the $\vec{u}$ of the particles in a single box, except averaged over all boxes. Next, the temperature is normalized by calculating the actual temperature with

$$T \simeq \frac{1}{2N} \sum_{i=1}^{N} m_i \left(v_{ix}^2 + v_{iy}^2 \right) \quad , \tag{2.9}$$

and multiplying each velocity by the factor $\sqrt{\frac{T_{target}}{T_{actual}}}$. Finally, any desired starting velocity for the system is directly added to the velocity of each particle. This is done after the calculation of the kinetic energy for the temperature, because the coherent motion of all particles is

equivalent to a Galilean transformation, and has no effect on internal temperature. This could be done, for example, to have a flow in a channel (Section 2.5.5) with slip walls (Section 2.4). If it was applied before the temperature normalization, it would artificially cool the system.

2.4 Boundary Conditions

Since free particles will eventually leave the field, it is necessary to somehow constrain them to the designated area. To deal with this, boundary conditions are applied after the streaming step. There are a few types of boundary conditions: periodic, slip, no-slip, and thermal.

2.4.1 Periodic Boundary Conditions

Periodic boundary conditions are an implementation of an infinite area. This is done by taking particles on one edge of the field and transporting them to the opposite edge. In order to be able to make measurements using the total displacement however, it is necessary to preserve that parameter and apply the boundary condition to another one. This is effected by assigning

$$x_{simulation} = x - L * \lfloor x \rfloor \quad . \tag{2.10}$$

The reason why $x_{simulation}$ is separate from x is so that while all simulation interactions use the simulation coordinate, measurements can be done on the system which require knowing total displacement. For example, when measuring diffusion, if a particle went a distance L in one direction, $x_{simulation}$ would report that it had not moved, while x would show that it had. An example of periodic boundary conditions in one dimension is shown in Figure 2.4.

This is a simple boundary method that strictly conserves momentum and energy. Since the wrapping mimics an infinite space (topologically it is actually the surface of a toroid), it is also good for tests. It does, however, have the disadvantage of allowing the effects of a particle to propagate all the way around the simulation and come back to interact with the same particle. For small system sizes, such self-interactions can cause problems.

2.4.2 Slip and No-Slip Walls

Slip and no-slip walls are very similar, in that they both take an incoming particle and reverse its direction in the direction normal to the wall. The difference is that while a slip

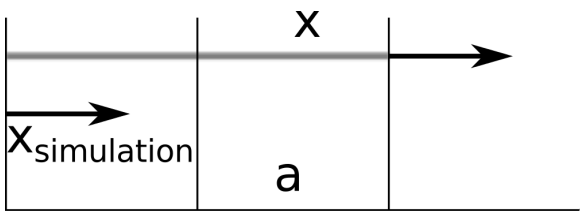
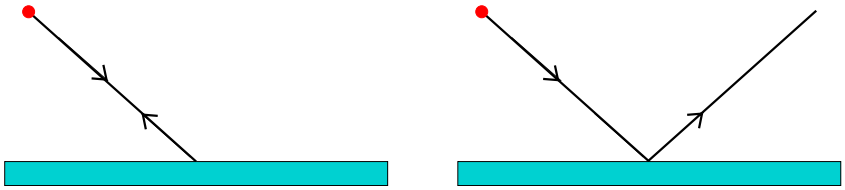


Figure 2.4: An example of one dimensional periodic boundary conditions.



(a) No-slip (b) Slip

Figure 2.5: Slip and no-slip walls.

wall does nothing to the tangential component of the velocity, a no-slip wall inverts it. To deal with the fact that the particle will have a position outside of the field, it is reflected back inside. This makes the particle act as if it had participated in the collision part way through the timestep, and had finished the available time with its new parameters. Both types of walls strictly conserve energy, but only conserve momentum on average.

2.4.3 Thermal Walls

Thermal walls entirely reset a particle's velocity in both directions. The normal component of the velocity, $v_\perp$, is drawn from a biased exponential distribution

$$P\left(v_\perp\right) = \frac{m v_\perp}{k_B T} e^{-\frac{m v_\perp^2}{2 k_B T}} \quad , \tag{2.11}$$

while the tangential component of the velocity, $v_{||}$, is drawn from a normal distribution

$$P\left(v_{||}\right) = \sqrt{\frac{m}{2\pi k_B T}} e^{-\frac{m v_{||}^2}{2 k_B T}} \quad . \tag{2.12}$$

This acts both as a no-slip boundary, as well as a thermostat at the wall [11].

For all wall types, the actual collision of the particle and the wall does not happen precisely at the end of a timestep. For the previous types of walls, this could be accounted for simply with algebra. However, in the case of thermal walls, the post-collision velocity is not a function strictly of the pre-collision velocity, and thus extra care is required. To deal with for this, the collision is re-calculated, with the timestep split into two parts. The intersection time of the particle and the wall is calculated, and the timestep is broken into two portions at that moment. The first portion of the timestep is used to bring the particle right up to the wall, and then the particle's velocity is assigned from the above distributions. Now that the particle has its new trajectory, it uses the second portion of the timestep to continue away from the wall.

2.4.4 Ghost Particles

One of the problems that arises when shifting and walls are used in conjunction with each other is that of shifting across a boundary. If one is not careful, shifting can make the system act (during the collision step) as if it had periodic boundary conditions, when it in fact does

not. This is because the shift could wrap one edge of the domain onto another, thus resulting in particles on one edge colliding with particles on the other edge. This could be avoided by making sure that they are included in separate boxes, as if on a domain of size L, with walls at a and $L - a$. This would result in partially filled collision boxes, however. To solve this, one may put a set of "ghost particles" into that box in order to make the box have M particles to collide. The velocities of these particles will be drawn from a Maxwell-Boltzmann distribution, with zero mean velocity and the same temperature as the wall [11].

2.5 Equilibrium Measurements

There are a number of coefficients that can both be calculated from the properties of the SRD model and can be experimentally measured from the simulation. In order for the simulation to be valid, both the experimental and theoretical values of these must match.

2.5.1 Momentum and Energy

The easiest parameters to measure are momentum and energy. Without the influence of walls or external forces, the two quantities should never change, as shown in Section 2.2.3. As demonstrated by Figures 2.6, 2.7, and 2.8, energy and momentum are constant at their expected values. This means that they are conserved by the SRD algorithm, as expected.

2.5.2 Velocity Distributions

While the total momentum remains constant, the distribution of velocities can vary as the simulation runs. While the particles are initialized with uniformly distributed velocities, their distribution should change to match the Maxwell-Boltzmann distribution,

$$P\left(v_\alpha\right) dv_\alpha = \sqrt{\frac{m}{2\pi k_B T}} e^{-\frac{m}{2k_B T} v_\alpha^2} dv_\alpha \quad , \tag{2.13}$$

where k_B is the Boltzmann constant, T is the temperature once the system reaches equilibrium [21], and α is a direction x, y, or z.

As shown in Figures 2.9 and 2.10, the velocity distributions quickly converge to the form given by the Maxwell-Boltzmann distribution. Since the simulation was initialized with a uniform velocity distribution, this shows that the SRD method correctly generates its equilibrium distribution.

2.5.3 Pressure

In a thermodynamically accurate simulation of an ideal gas (what SRD models), the ideal gas law $PV = Nk_BT$, where P is pressure, V is volume, and N is the number of particles should be obeyed. However, since the simulation is two dimensional, the corresponding equation is

$$PA = Nk_BT \quad , \tag{2.14}$$

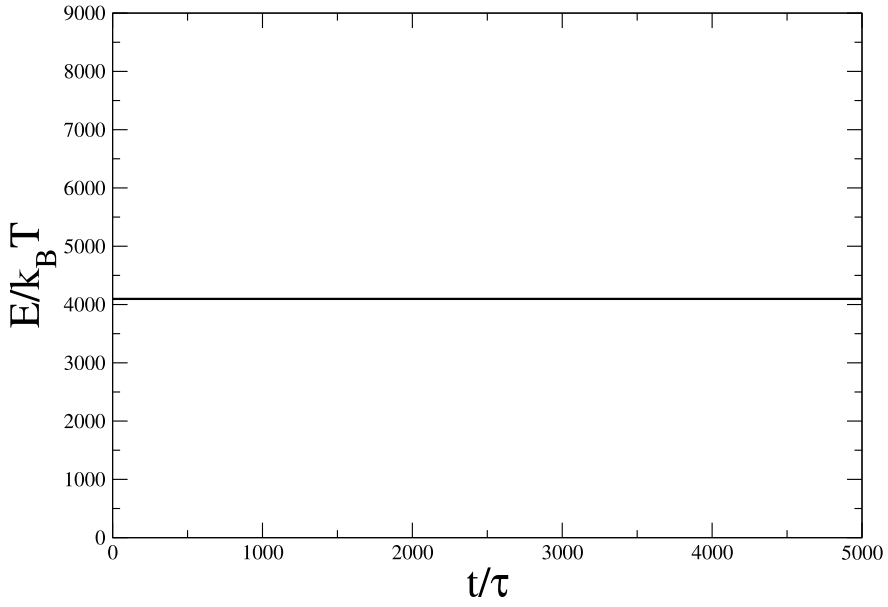


Figure 2.6: Total energy with respect to time. $L = 32$, $a = 1$, $N = 4096$, $M = 4$, $k_BT = 1$, $\tau = 1$, $\alpha = 90°$.

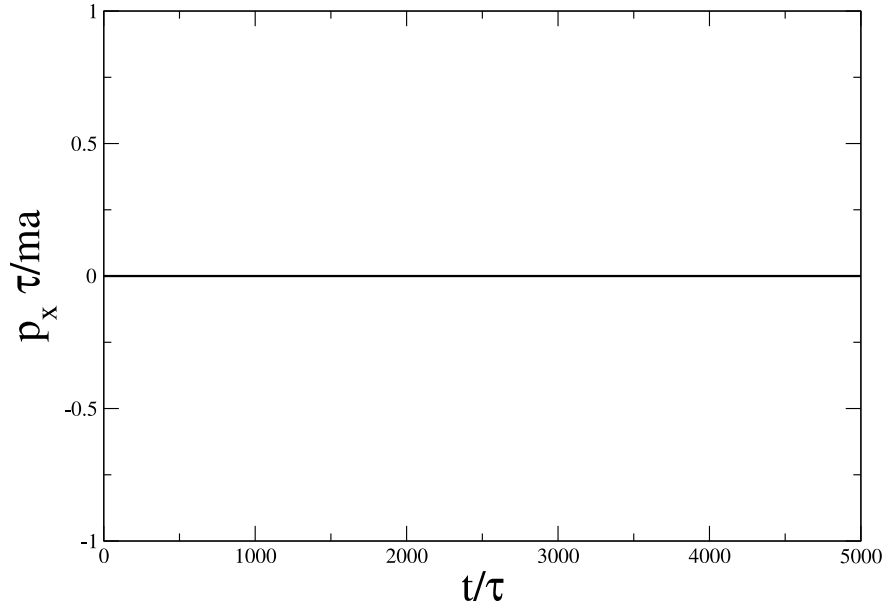


Figure 2.7: Total x momentum with respect to time. $L = 32$, $a = 1$, $N = 4096$, $M = 4$, $k_BT = 1$, $\tau = 1$, $\alpha = 90°$.

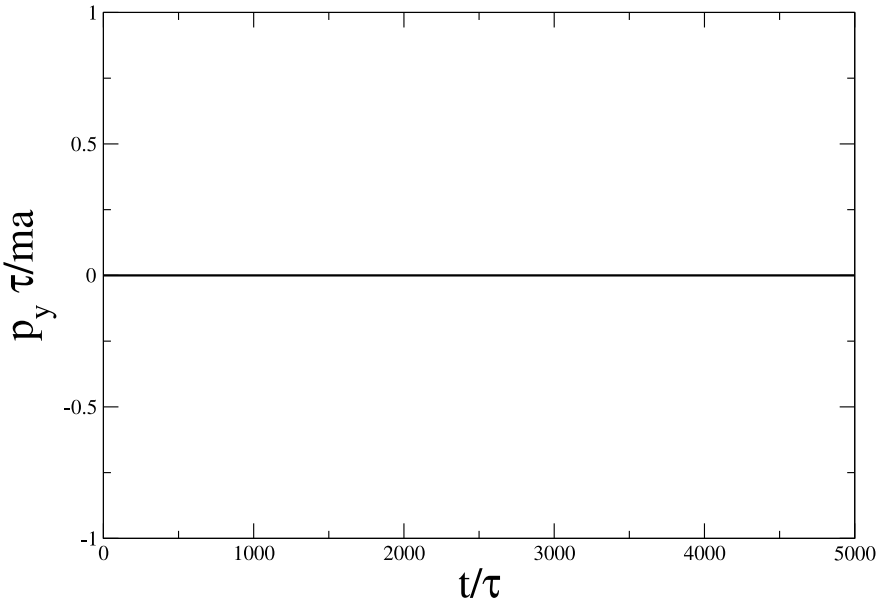


Figure 2.8: Total y momentum with respect to time. $L = 32$, $a = 1$, $N = 4096$, $M = 4$, $k_B T = 1$, $\tau = 1$, $\alpha = 90°$.

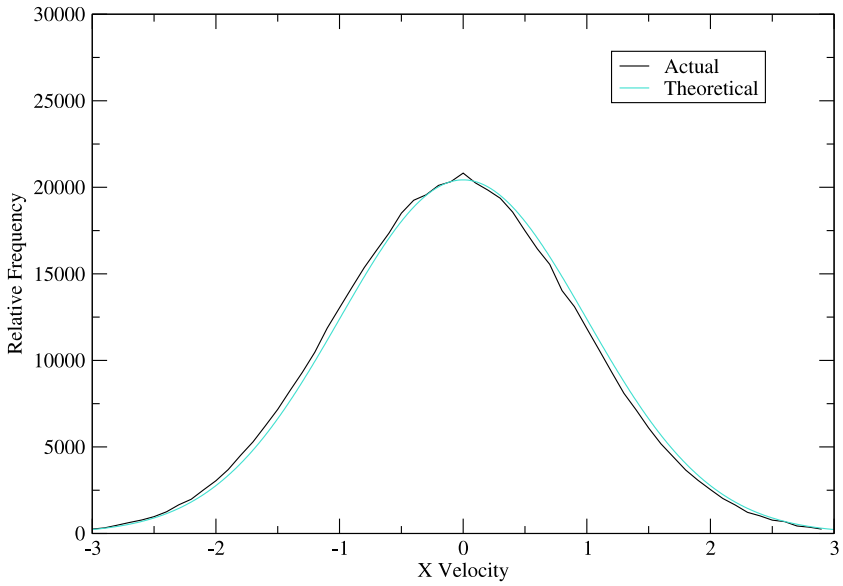


Figure 2.9: Distribution of x velocities at the end of a simulation run. $a = 1$, $\tau = 1$, $k_B T = 1$, $\alpha = 90°$.

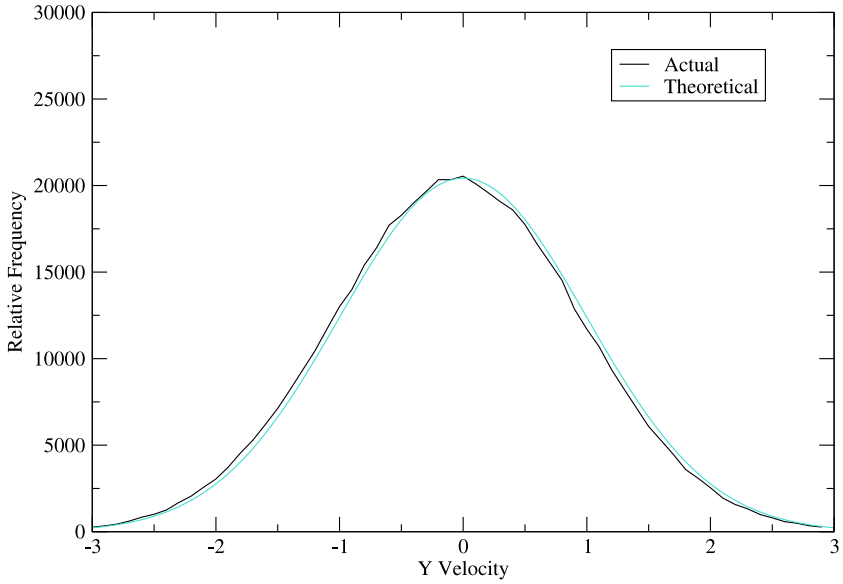


Figure 2.10: Distribution of y velocities at the end of a simulation run. $a = 1$, $\tau = 1$, $k_B T = 1$, $\alpha = 90°$.

where P is linear pressure (force per unit length) and A is the area. Pressure is determined based on the momentum traveling through a line. Instead of measuring the force on the line, the simulation allows one to measure the force that would have been on it, without actually having to affect the fluid.

Since all of the parameters except for P can be set for the simulation, it is easy to test this. Figures 2.11 and 2.12 show pressure as a function of area and temperature respectively.

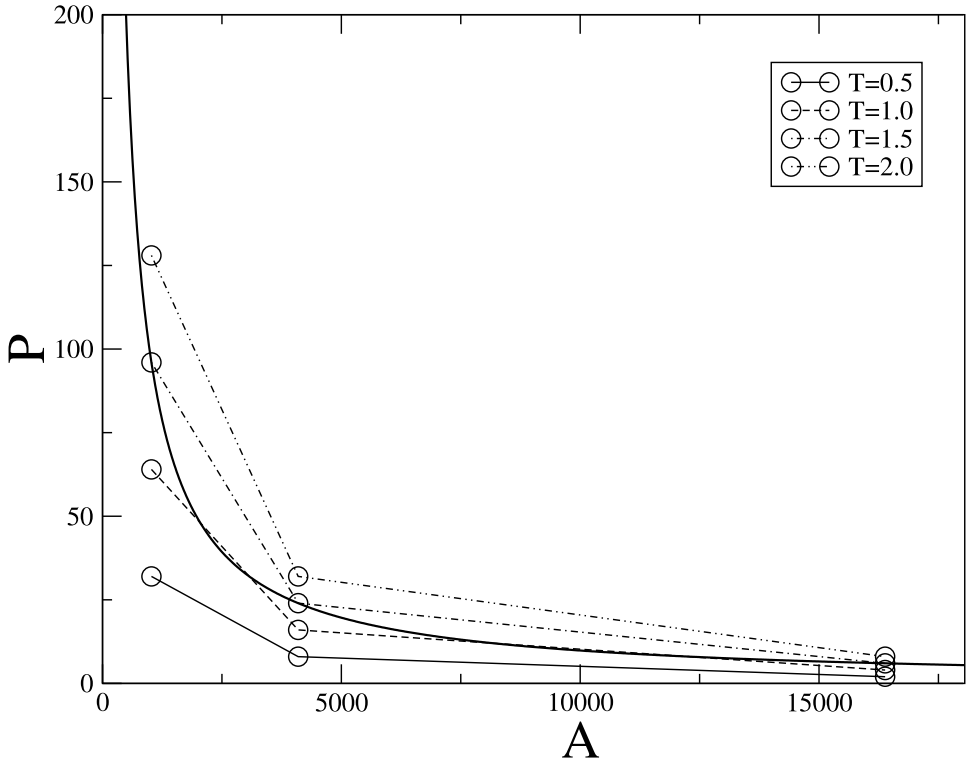


Figure 2.11: Pressure as a function of area for different values of temperature. $N = 65,536$, $a = 1$, $\tau = 1$, $\alpha = 90°$.

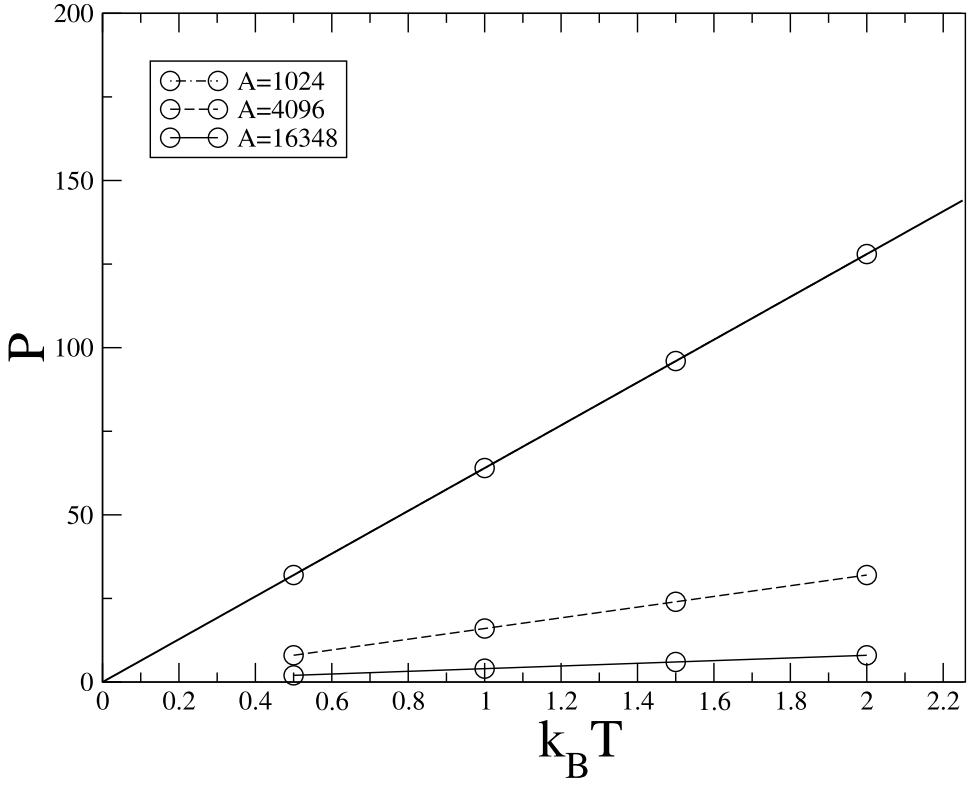


Figure 2.12: Pressure as a function of temperature for different values of area. $N = 65,536$, $a = 1$, $\tau = 1$, $\alpha = 90°$.

As predicted, pressure and area have an inverse relation, while pressure and temperature are directly proportional. This means that the simulation is properly modeling an ideal gas.

2.5.4 Diffusive Transport

Another parameter that can be measured is the diffusion coefficient D. This is measured as change in the mean squared displacement of the particles as a function of time, i.e.

$$\langle x^2 \rangle = 2Dt \ , \tag{2.15}$$

where x is the coordinate of a particle. Analytical calculation shows that the diffusion coefficient should work out to [21]

$$D = \frac{k_B T \tau}{2M} \left(\frac{2M}{(1 - \cos\alpha)\,(M - 1 + e^{-M})} - 1 \right) \ , \tag{2.16}$$

where τ is the timestep size, M is the number of particles per cell (Equation (2.1)), and α is the angle rotated through during a collision (Equation (2.6)). This coefficient has no dependence on direction, its value should be the same for both dimensions. One would expect that in a finite simulation domain, diffusion could only be measured for a short period of time until the particles under consideration run into the edges of the simulation. This happens quite quickly (on the order of tens to hundreds of timesteps depending on parameters). In order to properly measure diffusion for long periods of time, periodic boundary conditions are used (Section 2.4). Due to the wrapping explained earlier, the simulation domain appears infinite. This means that distances far larger than the available space can be observed, and the diffusion coefficients can be calculated over a longer time (and thus more accurately). The larger diffusion area allowed is shown in Figure 2.13.

The diffusion coefficient was measured for a sample of particles whose locations at the start of the simulation were recorded, and differences averaged at each timestep (Figure 2.14). The expected line from Equation 2.16 is plotted along with both the x and y diffusion results. The diffusion coefficients are the slopes of the lines. The agreement is excellent.

Before shifting was implemented, the simulation had Galilean invariance problems. When the particles are at a low enough temperature or high enough density, and the system has a flow, the diffusion coefficient in the direction of motion was greater than the diffusion coefficient in the other direction (Figure 2.15). This was solved by implementing shifting as described in Section 2.2.4. The improved version behaves as expected regardless of flow.

2.5.5 Viscous Transport

Another parameter that can be obtained from the simulations is the viscosity. This can easily be measured by applying thermal walls to one dimension of the simulation, emulating an infinite channel. By including an acceleration term, a flow can be induced in the fluid. This is usually called Poiseuille flow, and has a parabolic velocity distribution given by

$$v(x) = -\frac{\rho g}{2\eta} \left[\left((x - \frac{L}{2} \right)^2 - \left(\frac{L}{2} \right)^2 \right] \tag{2.17}$$

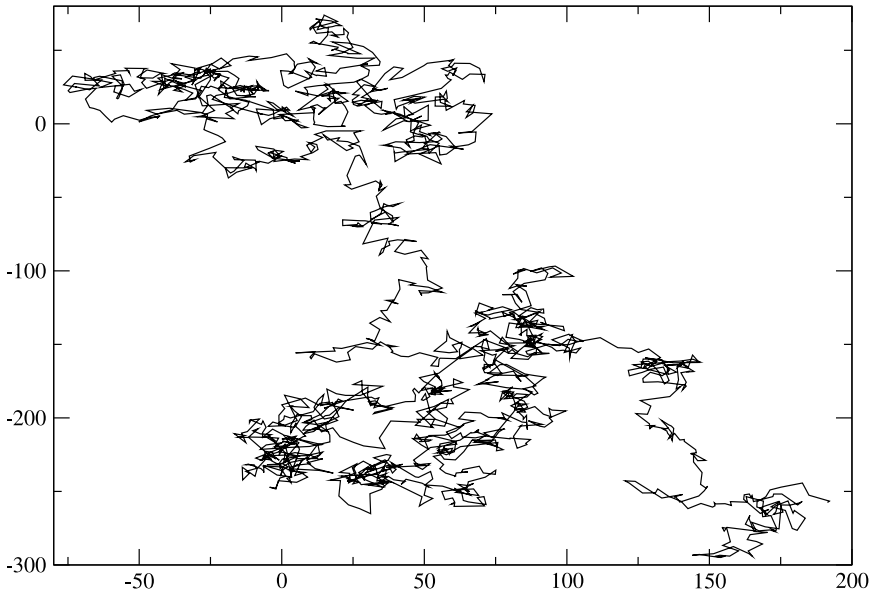


Figure 2.13: Traced path of one particle over 20,000 timesteps. $L = 32$, $a = 1$, $T = 1$, $\tau = 1$, $\alpha = 90°$.

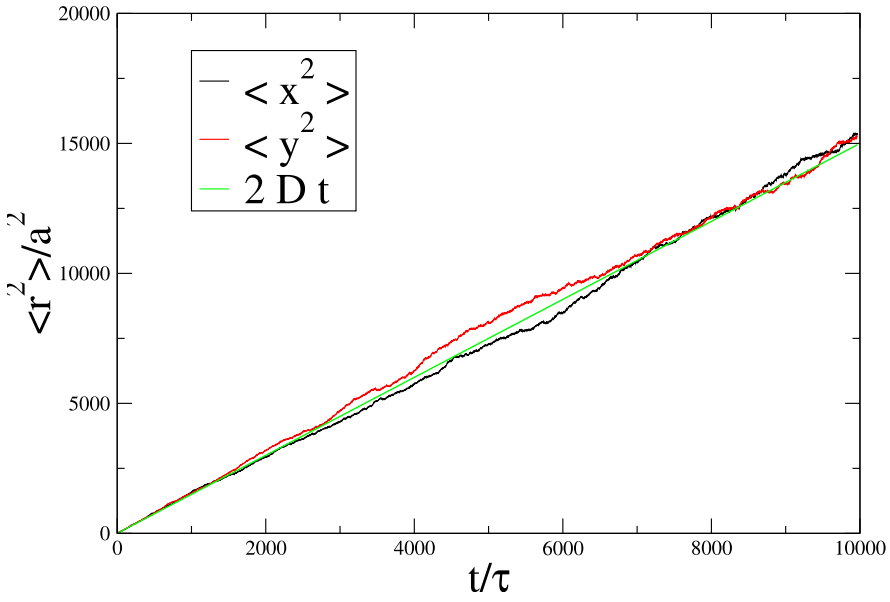


Figure 2.14: Mean squared displacement as a function of time for x and y components. Solid line is the theory from Equation (2.16).

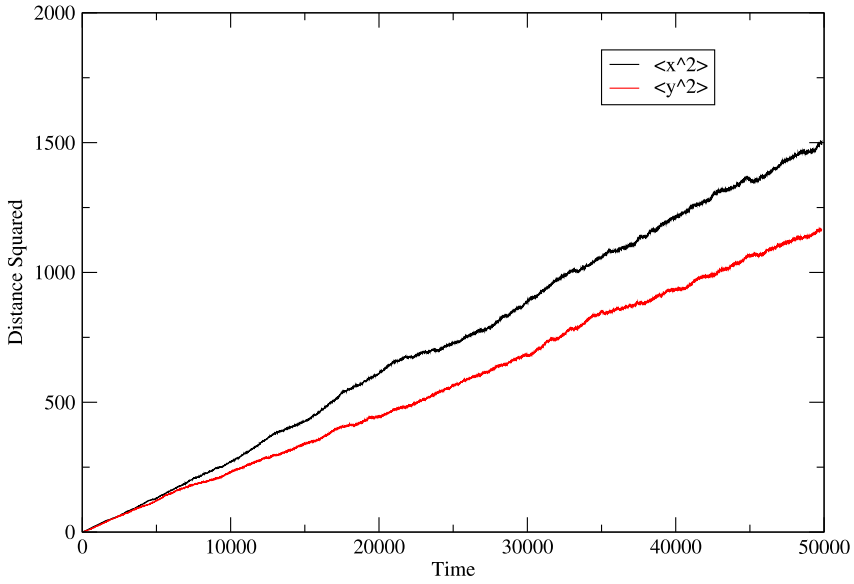


Figure 2.15: Diffusion when a flow is imposed in the x direction. The diffusion coefficients differ by 20% between the x and y directions when shifting is not enabled. $M = 16$, $T = 0.015625$, $\tau = 1$, $\alpha = 90°$.

where $\rho = mM = m\frac{N}{L^2}$ is the density and g is the acceleration of each particle.

This can be shown by starting with the definition of viscosity, which is

$$F = \eta \frac{\partial v}{\partial x} \ , \tag{2.18}$$

where F is the force, v is the velocity, and x is the position. Taking a 2D channel from $-L/2$ to $L/2$, the system will be symmetric across $x = 0$. One can thus treat the channel as a smaller channel from 0 to $L/2$, with the side at $x = 0$ frictionless. In equilibrium, the forces will balance, and so a differential segment of fluid at x will support all of the fluid from $x = 0$ to x. Equivalently, it will support

$$F = g\rho x = \eta \frac{\partial v}{\partial x} \ , \tag{2.19}$$

which leads to

$$\frac{\partial v}{\partial x} = \frac{\rho x}{\eta} \ . \tag{2.20}$$

Integrating from 0 to x and shifting so that $v(L/2) = 0$ gives

$$v(x) = \frac{\rho g}{\eta} \frac{1}{2} \left[\left(\frac{L}{2} \right)^2 - x^2 \right] \ .$$

A transformation from $x = 0$ to $x = L/2$ gives Equation (2.17).

This means that the experimental result can be matched against the expected profile to extract the viscosity η. The viscosity can also be calculated using

$$\eta = \frac{k_B T \tau}{2m} \left(\frac{M}{(M - 1 + e^{-M}) \sin^2 \alpha} - 1 \right) + \frac{a^2}{\tau} \frac{1}{6dM} \left(M - 1 + e^{-M} \right) (1 - \cos \alpha) \tag{2.21}$$

where a is the width of each box [21].

With the Poiseuille flow, the velocity distribution, shown in Figure 2.16, was parabolic as expected from Equation (2.13). The temperature distribution across the channel was approximately constant across the middle of the channel, also as expected (Figure 2.17).

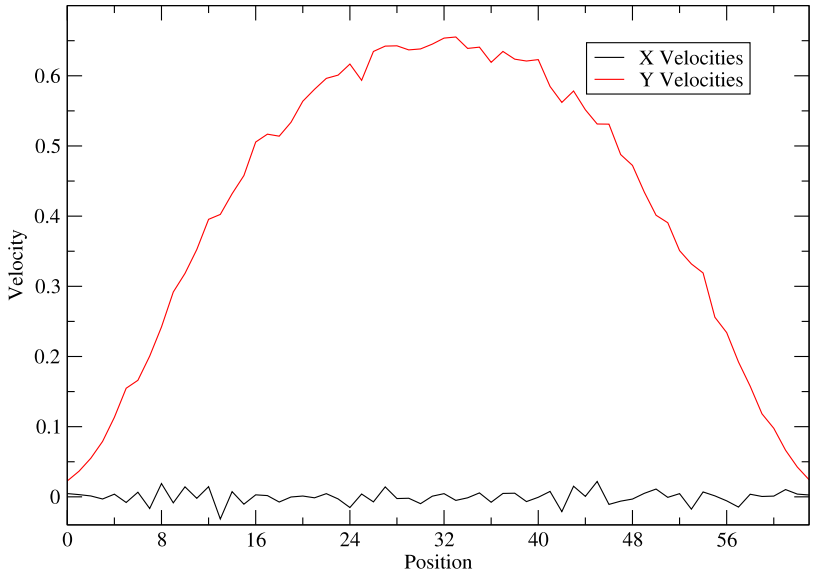


Figure 2.16: Velocity distribution across a channel. $a = 1$, $k_BT = 0.01$, $L = 64$, $\tau = 1$, $m = 1$, $M = 4$, $g = 0.0001$, $\alpha = 90°$

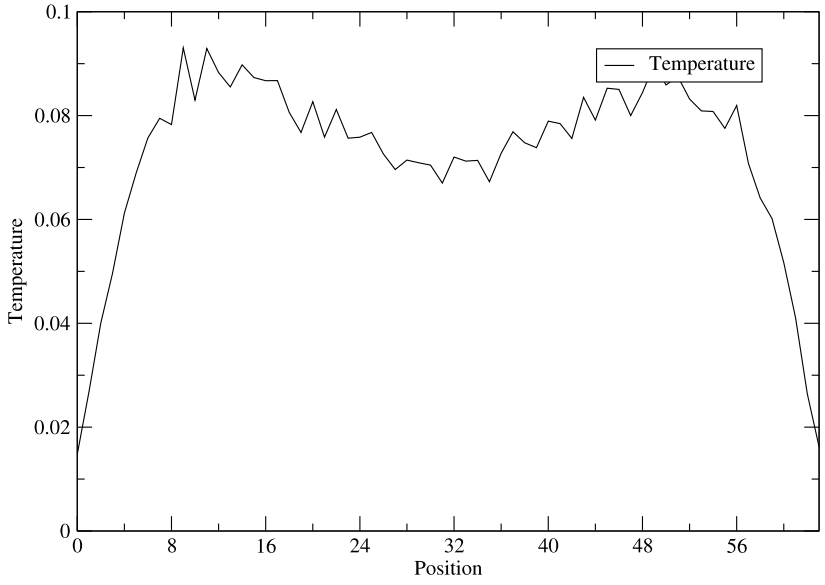


Figure 2.17: Temperature distribution across a channel. $a = 1$, $k_BT = 0.01$, $L = 64$, $\tau = 1$, $m = 1$, $M = 4$, $g = 0.0001$, $\alpha = 90°$.

Chapter 3

Modeling Polymers in an SRD Solvent

The coarse-grained nature of the SRD simulation method allows it to be used as a solvent for other methods. Taking advantage of this, a molecular dynamics module was created and used to model polymers. This chapter explains the methods used for modeling a polymer. Once modeled, applications range from viscous relaxation to swimming organisms.

3.1 Modeling of Polymers

Since polymers are long repeating chemical chains, they can be modeled easily as a chain of points connected together. These connections can be modeled as potentials. A cartoon description of a polymer is given in Figure 3.1.

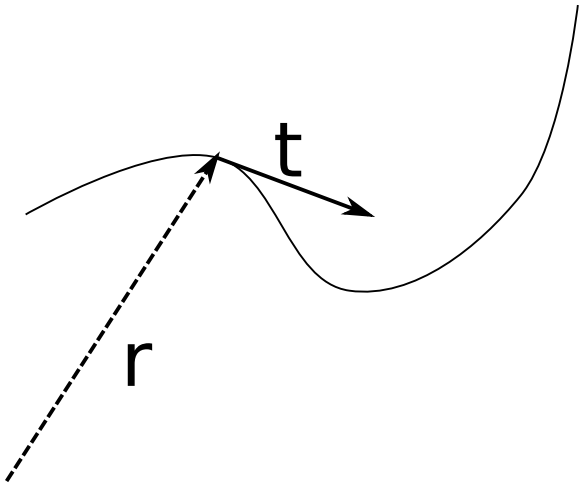


Figure 3.1: An example of a continuous polymer.

3.1.1 Continuum Model

In order to describe the potentials that describe a polymer, one may first consider a continuum model. Using $\vec{r}(s)$ as the position of each point on the polymer, and

$$\vec{t}(s) = \frac{\partial \vec{r}}{\partial s} \tag{3.1}$$

as the tangent vector to $\vec{r}$ is a good way to start. If $\partial\vec{t}/\partial s = 0$, $\vec{t}(s)$ is constant, and thus the polymer is a straight rod. Thus, bending energy must be a quadratic of $\partial\vec{t}/\partial s$. By definition ($\vec{t}$ is a unit tangent vector),

$$\vec{t} \cdot \frac{\partial \vec{t}}{\partial s} = 0 \tag{3.2}$$

and so the next available term is $\left(\frac{\partial \vec{t}}{\partial s}\right)^2$. This gives

$$U_{bend} = \frac{1}{2} EI \int_0^{L'} ds \left(\frac{\partial \vec{t}}{\partial s}\right)^2 \tag{3.3}$$

where L' is the length, and $EI = L_p k_B T$ is controlled by L_p, the persistence length. From statistical mechanics, the distribution of angles will follow a canonical distribution

$$\begin{aligned} \psi(U) &\propto e^{-\frac{U}{k_B T}} \\ &\propto e^{-\frac{IE}{2k_B T}\int_0^{L'} ds \left(\frac{\partial \vec{t}}{\partial s}\right)^2} \\ &\propto e^{-\frac{L_p}{2}\int_0^{L'} ds \left(\frac{\partial \vec{t}}{\partial s}\right)^2} \end{aligned}$$

From here, one can show that

$$\langle \left(\vec{t}(s) - \vec{t}(0)\right)^2 \rangle = \frac{2s}{L_p} \ , \tag{3.4}$$

which is similar in form to the diffusion equation (2.15). Thus, L_p^{-1} acts like a diffusion coefficient for the polymer.

End-to-end distance R_{ee} can also be calculated, i.e.,

$$\langle R_{ee}^2 \rangle = \int_0^{L'} ds \int_0^{L'} ds' \langle \vec{t}(s) \cdot \vec{t}(s') \rangle \tag{3.5}$$

using $\langle \vec{t}(s) \cdot \vec{t}(0) \rangle = e^{-\frac{s}{L_p}}$ gives

$$\langle R_{ee}^2 \rangle = 2L'L_p - 2L_p^2 \left(1 - e^{-\frac{L'}{L_p}}\right) \tag{3.6}$$

As expected, this gives the correct behavior in its limits. With $\frac{L'}{L_p} \gg 1$, $\langle R_{ee}^2 \rangle = 2L'L_p$. For $\frac{L'}{L_p} \ll 1$, $\langle R_{ee}^2 \rangle = (L')^2$.

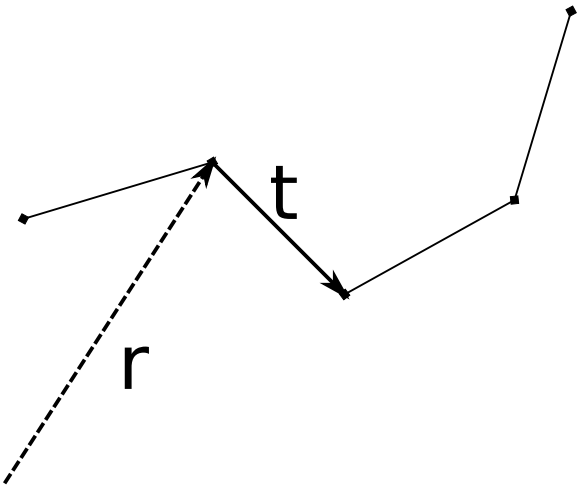


Figure 3.2: An example of a discretized polymer with four segments.

3.1.2 Discrete Treatment

In order to simulate the polymer, however, it needs to be broken down into discrete lengths, as shown in Figure 3.1.2. Making a discrete treatment of the polymer involves using $\vec{r}_i$, and $\vec{t}_i$, instead of $\vec{r}(s)$ and $\vec{t}(s)$, respectively. The bending energy can immediately be converted to

$$U_{bend} = \frac{1}{2} EI \sum_{i=1}^{n-1} \frac{\left(\vec{t}_{i+1} - \vec{t}_i\right)^2}{\left(\vec{r}_{i+1} - \vec{r}_i\right)^2} l \quad . \tag{3.7}$$

Here l is the segment length and n is the number of segments. Since $\vec{t}_i$ is a unit vector, combined with subtraction of zero-point energy, the numerator of the sum reduces to $-2\vec{t}_i \cdot \vec{t}_{i+1}$. Assuming that segments have fixed length, $\left(\vec{r}_{i+1} - \vec{r}_i\right)^2 = l^2$. Thus,

$$U_{bend} = -\frac{\kappa}{l} \sum_{i=1}^{n-1} \vec{t}_i \cdot \vec{t}_{i+1} \tag{3.8}$$

where $\kappa \equiv EI$. This means that the polymer chains can be given arbitrary rigidities, controlled by κ.

3.1.3 Stretching Energy

For the component of the energy from stretching, the discrete model can be used from the beginning. Placing a spring between each point on the chain gives a potential of,

$$U = 1/2k \left(|\vec{r}_{i-1} - \vec{r}_i| - l\right)^2 \tag{3.9}$$

such that the forces obeyed Hooke's law with spring constant k.

$$\vec{F} = -k \left(|\vec{r}_{i-1} - \vec{r}_i| - l\right) \hat{t} \tag{3.10}$$

This constraint makes the polymer behave as a freely jointed chain. Each individual link is kept at its given length l, and the links can rotate freely.

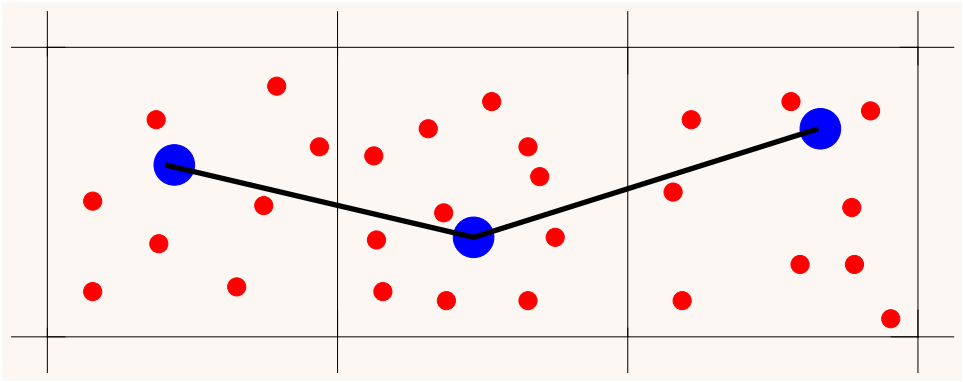


Figure 3.3: Coupling between the fluid and a polymer.

3.1.4 Resulting Forces

The total internal potential energy for the polymer is then given by

$$U_{internal} = U_{bend} + U_{stretch} \quad . \tag{3.11}$$

Thus, the behavior of the polymer is

$$m\frac{d^2x}{dt^2} = \nabla U_{internal} + F_{random} + F_{dissipative} \tag{3.12}$$

where $U_{internal}$ are the internal potentials, and there are an additional two forces: F_{random} and $F_{dissipative}$. The random force applies Brownian kicks to the polymer, while the dissipative force acts as a heat bath to absorb energy from the polymers relaxation. Unless solvent molecules are included explicitly these forces do not conserve momentum, and therefore do not produce the correct in hydrodynamic behavior.

3.1.5 SRD Heat Bath

It is possible to have molecular dynamics support full hydrodynamics, but it requires modeling the solvent as well, which is very expensive.

However, it is possible to use an inexpensive SRD solvent as the heat bath for the polymer. The SRD solvent can provide the friction and random force required for the MD to work properly. Since the forces are the product of interactions with the surrounding fluid, momentum is conserved, and is passed back to the fluid.

The coupling is done by having the particles that are part of MD participate in the SRD collisions as if they were regular solvent particles. This gives

$$\vec{u} = \frac{\sum_{i=0}^{M'} m_i\vec{v}_i + \sum_{j=0}^{M''} m_j\vec{v}_j}{\sum_{i=0}^{M'} m_i + \sum_{j=0}^{M''} m_j} \tag{3.13}$$

where the first sum (over i) is the same as in Equation (2.8), and the second sum (over j) is for the additional M'' particle(s) from the polymer.

These particles behave almost exactly the same way as the regular SRD particles, with one exception. Instead of moving normally as described in Section 2.2.1, they move based on the molecular dynamics calculations. These are done using a Velocity-Verlet algorithm.

3.1.6 Velocity-Verlet Algorithm

In order to get very rigid rods and joints, large spring constants are required. This works, though when the forces are are too strong, the simulation does not converge with reasonably sized timesteps. Since strong springs are essential to the algorithm working, other options are required to ensure convergence. The simplest way to improve this is to just use a smaller timestep. As such, the MD calculations were done with a timestep on the order of 1:100 the size of the SRD calculations.

Even with this, convergence is a bit of a problem. In order to help them converge faster, a velocity integration method with a lower error term is helpful. For this, velocity-corrected Verlet integration was used:

$$\vec{r}\left(t+\Delta t\right) = \vec{r}\left(t\right)+\vec{v}\left(t\right)\Delta t+\frac{1}{2}\vec{a}\left(t\right)\Delta t^{2} \tag{3.14}$$

$$\vec{v}\left(t+\Delta t\right) = \frac{1}{2}\left(\vec{a}\left(t\right)+\vec{a}\left(t+\Delta t\right)\right)\Delta t \tag{3.15}$$

Since the particles in the polymer chains are using a smaller timestep, they spend their movement time doing MD instead of the normal Eulerian streaming described previously.

3.2 Equilibrium Measurements

3.2.1 Overall Length

The simplest measure of a polymer is the end to end length. This is simply defined as

$$R_{ee} = |\vec{r_n} - \vec{r_0}| \tag{3.16}$$

In the case of a polymer with $\kappa = 0$, the points along the polymer chain follow a random walk. As such, the total length will follow a path similar to that a single point takes (Equation (2.15)). This means that the total length $L_t \propto \sqrt{n}$. However, when the polymer has a $\kappa \neq 0$, it follows a more straight path, and as κ approaches infinity, $R_{ee} \propto L'$. This is shown in Figure 3.4, by how the $\kappa = 0$ curve follows $\sqrt{n}$.

3.2.2 Radius of Gyration

The radius of gyration, R_g gives a measure of the dimensions of the chain. It is defined as

$$R_g^2 = \frac{1}{n}\sum_{i=0}^{n}\left(\vec{r} - \vec{r_{cm}}\right)^2 \tag{3.17}$$

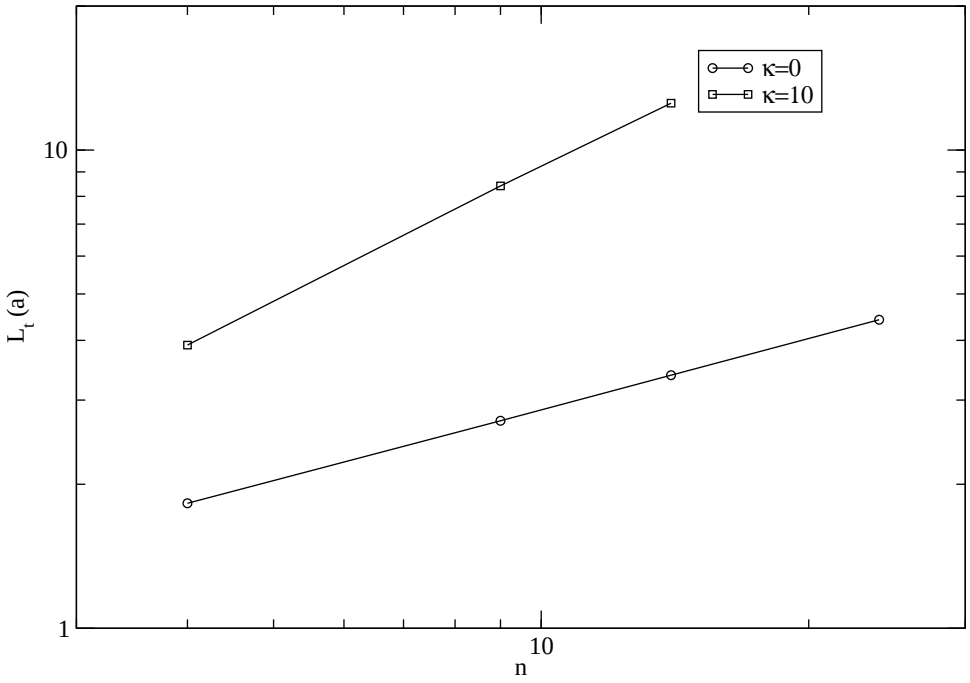


Figure 3.4: Polymer length for $\kappa = 0, 10$. $a = 1$, $l = 1$, $k_B T = 1$, $\tau = 1$, $L = 32$, $M = 5$.

where the center of mass is

$$\vec{r_{cm}} = \frac{1}{n} \sum_{i=0}^{n} \vec{r} \tag{3.18}$$

R_g depends on both the chain length n, and the stiffness, κ, as shown in Figure 3.5. A stiffer chain will be more spread apart, and thus have a greater R_g.

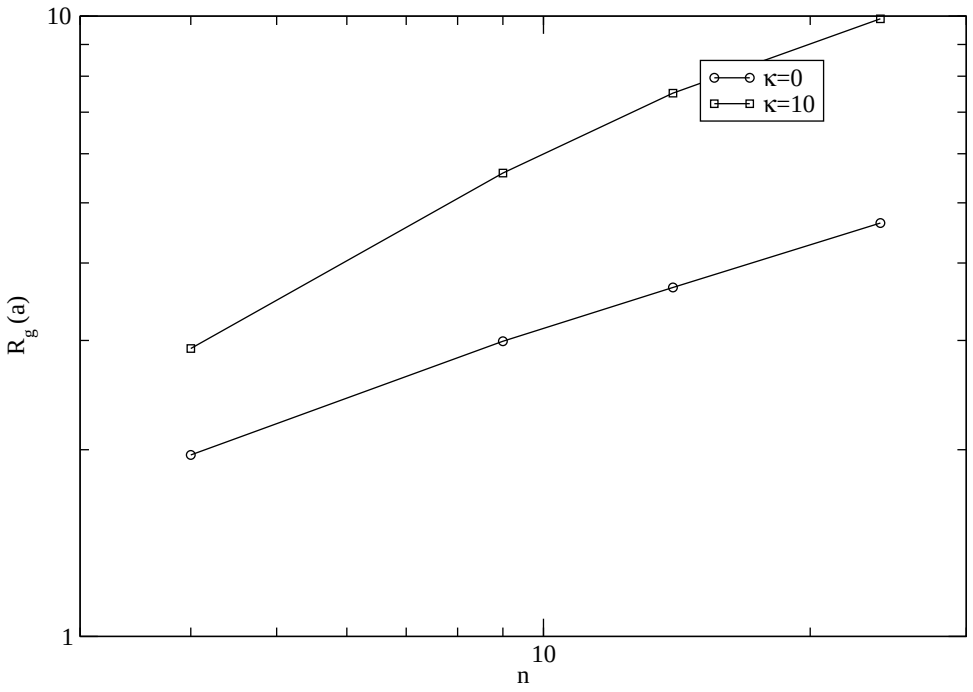


Figure 3.5: Radius of Gyration for $\kappa = 0, 10$. $a = 1$, $l = 1$, $k_B T = 1$, $\tau = 1$, $L = 32$, $M = 5$.

Chapter 4

Non-ideal and Binary Fluids

Since SRD uses point particles, the resulting equation of state is that of an ideal gas. In order to have a non-ideal equation of state, particles need to act as if they have non-zero diameter. This is done using a new type of collision rule, as explained in the next section.

4.1 Description of Model

The non-ideal model extends SRD by also including an inter-cell collision term. In each timestep, the collision step has an additional part, where all of the particles in a cell are collided against all of the particles in a neighboring cell.

4.1.1 Super-cells

In order to do the collisions, cell pairs are required. This is done by splitting the domain into super-cells of size $2a$, where a is the size of a SRD collision cell (Figure 4.1). In each super-cell, during each timestep, the particles can either collide horizontally, vertically, or diagonally. The probability of collisions is equal for each option. However, since there are two different ways of having horizontal collisions, it has twice the probability of the other two. Thus, horizontal and vertical collisions each have a 25% probability, while diagonal collisions have a 50% probability.

The super-cells used for non-ideal collisions also have shifting. The shifting used is similar to the shifting used in normal SRD (Section (2.2.4)), but it uses a range of $[-a, a]$. This is because the super-cells are of size $2a$, and spreading the shifting over that requires the greater range.

4.1.2 Modification of SRD Rules

All of the particles in a super-cell collide against all of the particles in the paired super-cell. The collision is simple, where the parallel components of the velocity interchange while the perpendicular components are unchanged. This gives a set of rules as described in Table 4.1.2.

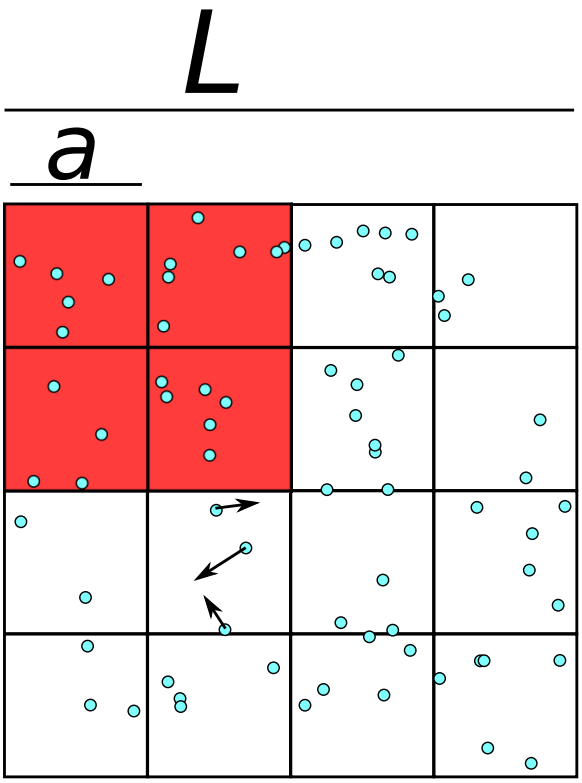
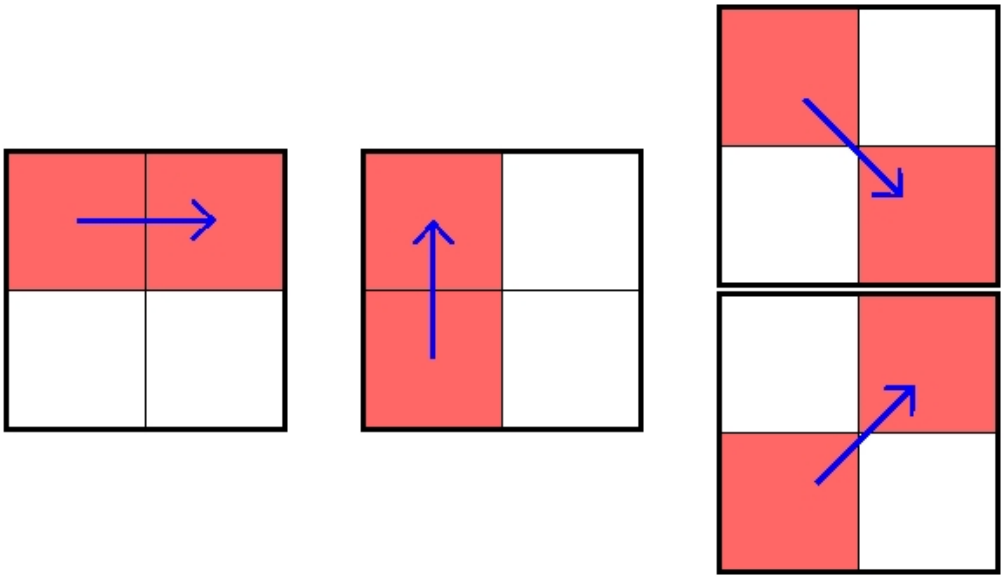


Figure 4.1: A super-cell on an SRD domain.



(a) Horizontal: $\vec{\sigma} = \hat{x}$ (b) Vertical: $\vec{\sigma} = \hat{y}$ (c) Diagonal: $\vec{\sigma} = \frac{\hat{x}+\hat{y}}{\sqrt{2}}, \frac{\hat{x}-\hat{y}}{\sqrt{2}}$

Figure 4.2: Possible collision pairs.

$\vec{\sigma}$	Collision		
$\hat{x}$	$v_x(t+\tau)$	$=$	$2u_x - v_x(t)$
	$v_y(t+\tau)$	$=$	$v_y(t)$
$\hat{y}$	$v_x(t+\tau)$	$=$	$v_x(t)$
	$v_y(t+\tau)$	$=$	$2u_y - v_y(t)$
$\frac{\hat{x}+\hat{y}}{\sqrt{2}}$	$v_x(t+\tau)$	$=$	$u_x + u_y - v_y(t)$
	$v_y(t+\tau)$	$=$	$u_x + u_y - v_x(t)$
$\frac{\hat{x}-\hat{y}}{\sqrt{2}}$	$v_x(t+\tau)$	$=$	$u_x - u_y + v_y(t)$
	$v_y(t+\tau)$	$=$	$u_y - u_x + v_x(t)$

Table 4.1: Non-ideal collision equations.

In order to have the proper thermodynamics, collisions happen with a probability P, given by

$$P = \tanh(A\Delta u M_a M_b) \quad , \tag{4.1}$$

where A is a parameter (like inverse temperature), Δu is the difference in velocities between box a and b, and M_a and M_b are the number of particles in each of the two participating boxes.

The hyperbolic tangent function is used because it provides a nice curve for mapping the domain $[0, \infty)$ onto $[0, 1]$. It is linear with slope $d\tanh(x)/dx = 1$ at $x = 0$, but asymptotically approaches 1 as x approaches infinity. This helps keep the probability in the appropriate domain, though A should be chosen such that the probability does not saturate. If A is too high, the probability looses its dependence on M_A, M_B, and Δu, which is a reflection of $k_B T$.

All of the different steps in a timestep should happen simultaneously in order to be invariant under time-reversal. For SRD this is not a problem, because the collision step does not affect position, and the streaming step does not affect velocity. For SRD and non-ideal collisions in the same system, however, this is important, and needs to be addressed. The simplest way of solving the problem is to randomly average it out, similar to the spacial random averaging used for shifting. By randomly picking which collision type to do first, it means that on average the collisions happen at the same time, and time-reversal invariance is restored.

4.2 Test of Model

4.2.1 Conservation Laws

The primary test of the new collision rules is conservation of momentum and energy. In theory both are conserved, since the collisions are elastic. Simulation results show that they are indeed conserved, as expected (data not shown).

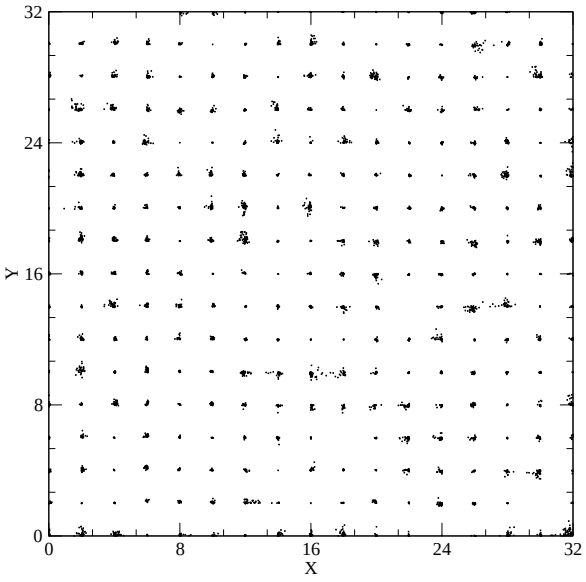


Figure 4.3: A frozen non-ideal fluid. $L = 32$, $a = 1$, $M = 5$, $k_BT = 0.001$, $\tau = 0.1$, $\alpha = 90°$. Though not thermodynamically consistent, A is chosen extremely large to emphasize the effect.

4.2.2 Freezing

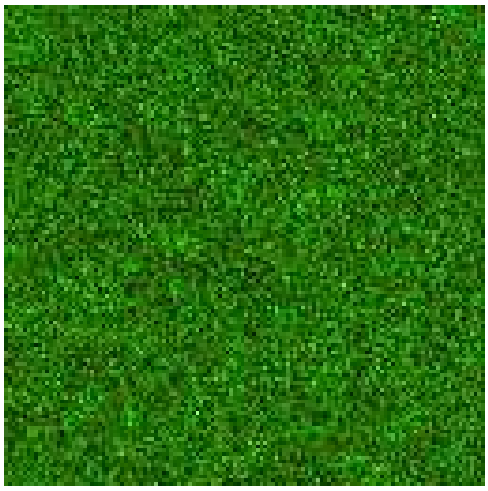
Since the new collisions model a hard-sphere fluid with a non-ideal equation of state, it is capable of freezing at low enough temperatures. The square structure of the SRD model means that the frozen fluid has a square crystal structure. When it freezes, sets of particles clump together. These particles behave as normal SRD, since they are close enough that the non-ideal collision rarely affects them (only when shifting causes a boundary to cross the group). The groups then collide with each other via the non-ideal term, since they are too far away for SRD to affect them. This keeps them stable.

4.3 Binary Fluid

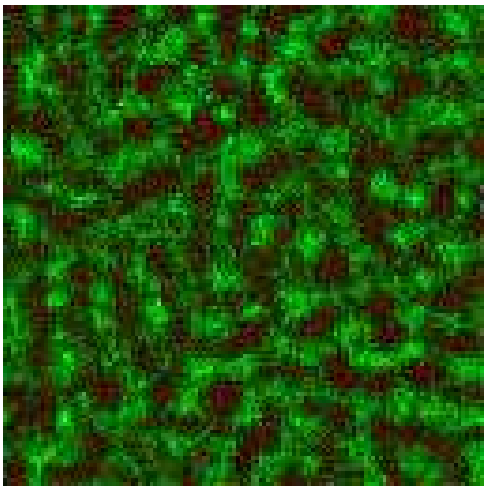
Using the non-ideal fluid collisions, a binary fluid can also be simulated. Instead of having just a single species of particle in the simulation, one can have two of them. They interact with each other differently (although symmetrically), resulting in an immiscible binary fluid, as explained in the following sections.

4.3.1 Binary Collisions

While the non-ideal collisions normally happen with all the particles in two adjacent boxes, in the case of binary collisions, it is only between different particle types. With two particle species A and B, A particles in the first box collide with B particles in the second box, while B particles in the first box collide with A particles in the second box. This means that



(a) Spontaneously cooled homogeneous mixture.



(b) After some time, the fluid has partially separated into its components.

Figure 4.4: Binary mixture separating.

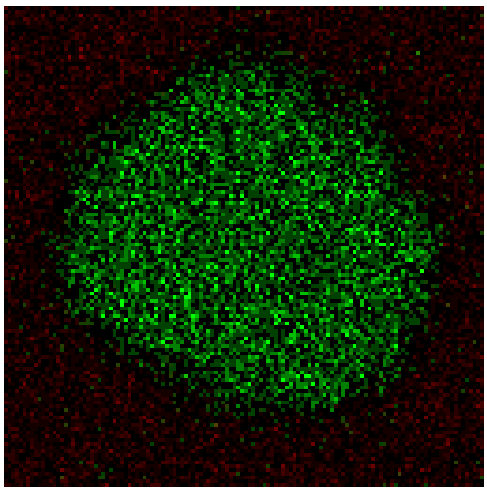


Figure 4.5: A binary droplet.

a homogeneous (only type A) fluid will behave as normal SRD. However, at an interface between A and B, the particles will collide with each other and be mutually repelled.

4.3.2 Phase Separation and Droplet Formation

The repulsion between the two species is effective enough to let the two fluids phase separate from a homogeneous mixture into areas of their components. Figure 4.3.2 shows 50%-50% mixture phase separating after starting from a completely random distribution. Figure 4.3.2 shows an example of a droplet of radius 12, which is made with a with a 56%-44% mixture.

Chapter 5

Conclusions and Future Directions

We wrote an implementation of the SRD fluid simulation technique. This implementation was analytically proven to conserve momentum an energy, and demonstrated to do so in practice as well. Previous work has analytically determined the expected diffusion coefficient and kinematic viscosity for an SRD fluid, and simulation results match these theoretical expectations. Additionally, pressure was measured, and the ideal gas properties of SRD were demonstrated.

This method has a number of potential applications. The first of these is combining molecular dynamics and multiple-component simulations to model surfactant systems. This would allow us to study capillary waves in the surface of a droplet, and look at how surface tension changes with the addition of a surfactant. Extending the molecular dynamics part would allow us to model a Purcell swimmer. While the Purcell swimmer has been modeled before in a continuum system, the Brownian kicks of the SRD solvent would provide a novel (and more realistic) environment. Depending on the implementation, it could be possible to have the rods connecting particles in molecular dynamics be solid boundaries. This would also enable the simulation of impermeable and semi-permeable membranes, commonly found in biological systems.

Appendix A

Code Samples

As our implementation of the SRD simulation method was written in C, the example pieces of code provided here are also.

A.1 Data Types

There are structs for particles and for boxes:

```
typedef struct {
        double sX; //position
        double sY;

        double vX; //velocity
        double vY;

        int rX; //wrap count
        int rY;
        int mass;
} particle;

typedef struct {
        double mX; //total momentum
        double mY;

        double count; //particles in the box
        int rotDir; //direction of rotation
} box;
```

A.2 SRD Collisions

The core of the SRD collision:

```
void collide(particle particles[], box boxes[], double boxShiftX, double boxShiftY) {
        int i;
        double nX;
        double nY;
        int boxnum;
        for(i=0;i<PART_COUNT;i++) {
```

```
                boxnum = CALC_BOX_NUM; //a macro for determining the ID of the box the particle is in.

                particles[i].vX -= boxes[boxnum].mX; //subtract average velocity
                particles[i].vY -= boxes[boxnum].mY;

                nX = particles[i].vX*Wxx+boxes[boxnum].rotDir*particles[i].vY*Wxy; //do rotation
                nY = boxes[boxnum].rotDir*particles[i].vX*Wyx+particles[i].vY*Wyy;

                particles[i].vX = nX + boxes[boxnum].mX; //add average velocity back
                particles[i].vY = nY + boxes[boxnum].mY;
        }
}
```

A.3 Eulerian Streaming

Simple Eulerian Streaming:

```
void move(particle particles[]) {
        int i;
        for(i=0;i<PART_COUNT;i++) {
                particles[i].sX += particles[i].vX*TAU;
                particles[i].sY += particles[i].vY*TAU;
        }
}
```

A.4 Thermal Walls

Thermal Walls:

```
inline void thermalWallsX(particle* part) { //inlined to improve efficiency
        double t;
        if(part->sX < 0) {
                t = part->sX/part->vX;
                part->sY -= t*part->vY;

                part->vX = randm()*SQRT_WALL_TEMP*SQRT_2;
                part->vY = randn()*SQRT_WALL_TEMP;

                part->sX = part->vX*t;
                part->sY += part->vY*t;
        } else if(part->sX > WIDTH) {
                t = (part->sX-WIDTH)/part->vX;
                part->sY -= t*part->vY;

                part->vX = -1*randm()*SQRT_WALL_TEMP*SQRT_2;
                part->vY = randn()*SQRT_WALL_TEMP;

                part->sX = WIDTH+part->vX*t;
                part->sY += part->vY*t;
        }
}
```

A.5 Periodic Boundary Conditions

```
inline void periodicBoundaryCondX(particle* part) {
        part->rX += floor(part->sX/WIDTH);
        part->sX -= WIDTH*floor(part->sX/WIDTH);
}
```

Bibliography

[1] F. J Alexander and A. L Garcia. The direct simulation monte carlo method. *Computers in Physics*, 11(6):588, 1997. 8

[2] G. A. Bird. *Molecular Dynamics and the Direct Simulation of Gas Flow*. Oxford Science Publications, Oxford, 1994. 8

[3] W. D Cornell, P. Cieplak, C. I Bayly, I. R Gould, K. M Merz, D. M Ferguson, D. C Spellmeyer, T. Fox, J. W Caldwell, and P. A Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *Journal of the American Chemical Society*, 117(19):5179–5197, 1995. 7

[4] E. R Davidson and D. Feller. Basis set selection for molecular calculations. *Chemical Reviews*, 86(4):681–696, 1986. 7

[5] J. K.G Dhont. *An introduction to dynamics of colloids*. Elsevier Science, 1996. 8

[6] P. Espanol. Hydrodynamics from dissipative particle dynamics. *Physical Review E*, 52(2):1734–1742, 1995. 7

[7] P. Espanol and P. Warren. Statistical mechanics of dissipative particle dynamics. *EPL (Europhysics Letters)*, 30:191, 1995. 7

[8] Robert Eymard, Thierry Gallouët, Raphaèle Herbin, P.G. Ciarlet, and J.L. Lions. Finite volume methods. In *Solution of Equation in [real]n (Part 3), Techniques of Scientific Computing (Part 3)*, volume Volume 7, pages 713–1018. Elsevier, 2000. 6

[9] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier-Stokes equation. *Physical review letters*, 56(14):1505–1508, 1986. 6

[10] Alejandro Garcia. *Numerical methods for physics*. Prentice Hall, Upper Saddle River N.J., 2nd ed. edition, 2000. 8

[11] G. Gompper, T. Ihle, D. Kroll, and R. Winkler. Multi-particle collision dynamics: A particle-based mesoscale simulation approach to the hydrodynamics of complex fluids. In Christian Holm and Kurt Kremer, editors, *Advanced Computer Simulation Approaches for Soft Matter Sciences III*, volume 221 of *Advances in Polymer Science*, pages 1–87. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-540-87706-6-1. 8, 17, 18

[12] X. He, S. Chen, and G. D Doolen. A novel thermal model for the lattice boltzmann method in incompressible limit* 1. *Journal of Computational Physics*, 146(1):282–300, 1998. 6

[13] P. J. Hoogerbrugge and J. Koelman. Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. *EPL (Europhysics Letters)*, 19:155, 1992. 7

[14] T. Ihle and D. Kroll. Stochastic rotation dynamics:a galilean-invariant mesoscopic model for fluid flow. *Physical Review E*, 63(2), 2001. 8, 14

[15] R. G Larson. The structure and rheology of complex fluids. *New York: Oxford*, 2001. 9

[16] Anatoly Malevanets and Raymond Kapral. Mesoscopic model for solvent dynamics. *The Journal of Chemical Physics*, 110(17):8605, 1999. 8, 10

[17] S. L Mayo, B. D Olafson, and W. A Goddard. DREIDING: a generic force field for molecular simulations. *Journal of Physical Chemistry*, 94(26):8897–8909, 1990. 7

[18] G. R McNamara and G. Zanetti. Use of the boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, 61(20):2332–2335, 1988. 6

[19] M. Ripoll, K. Mussawisade, R. G. Winkler, and G. Gompper. Dynamic regimes of fluids simulated by multiparticle-collision dynamics. *Physical Review E*, 72(1):016701, 2005. 8

[20] X. Shan and H. Chen. Lattice boltzmann model for simulating flows with multiple phases and components. *Physical Review E*, 47(3):1815, 1993. 6

[21] E. Tüzel, T. Ihle, and D. Kroll. Dynamic correlations in stochastic rotation dynamics. *Physical Review E*, 74(5), 2006. 18, 23, 26

[22] Frank White. *Viscous fluid flow*. McGraw-Hill Higher Education, New York NY, 3rd ed. edition, 2006. 9